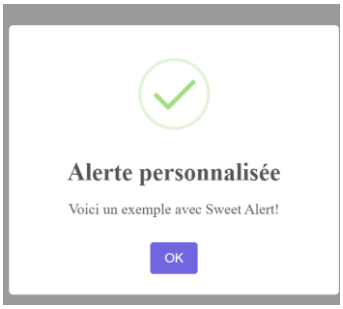
Exercice 1: Intégrer un composant simple

1. Intégrer le composant Sweet Alert sur une page web.
2. Personnaliser son aspect en modifiant le style des alertes via sa documentation.



Correction

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Sweet Alert</title>
  <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
</head>
<body>
  <button id="alertButton">Afficher l'alerte</button>
  <script>
    document.getElementById('alertButton').onclick = function () {
      Swal.fire({
        title: 'Alerte personnalisée',
        text: 'Voici un exemple avec Sweet Alert!',
        icon: 'success',
        confirmButtonText: 'OK'
      });
    };
  </script>
</body>
</html>
```

Exercice 2: Prototyper un carousel

1. Créer une page HTML avec un élément `<img>` pour afficher une image, et un `<button>` pour changer l'image.
2. Ajouter un tableau contenant trois URLs d'images.
3. Afficher la première image du tableau au chargement de la page.
4. Permettre à l'utilisateur de passer à l'image suivante en cliquant sur le bouton.

Correction

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Carousel Prototype</title>
</head>
<body>
  <img id="carouselImage" src="" alt="Carousel" style="width: 300px; height: 200px;">
  <button id="nextButton">Suivant</button>
  <script>
    const images = [
      'https://via.placeholder.com/300x200?text=Image+1',
      'https://via.placeholder.com/300x200?text=Image+2',
      'https://via.placeholder.com/300x200?text=Image+3'
    ];
    let currentIndex = 0;
    const carouselImage = document.getElementById('carouselImage');
    const nextButton = document.getElementById('nextButton');
    // Afficher la première image
    carouselImage.src = images[currentIndex];
    nextButton.onclick = function () {
      currentIndex = (currentIndex + 1) % images.length;
// Utilisation de l'opérateur modulo
      carouselImage.src = images[currentIndex];
    };
  </script>
</body>
</html>
```

Exercice 3: Créer un accordéon

1. Créer un composant accordéon avec trois sections.
2. Permettre à l'utilisateur d'ouvrir une section en cliquant sur son titre et de fermer les autres.

Correction

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Accordéon</title>
  <style>
    .section {
      border: 1px solid #ccc;
      margin: 5px;
    }
    .content {
      display: none;
      padding: 10px;
    }
  </style>
</head>
<body>
  <div class="section">
    <h3 class="title">Section 1</h3>
    <div class="content">Contenu de la section 1</div>
  </div>
  <div class="section">
    <h3 class="title">Section 2</h3>
    <div class="content">Contenu de la section 2</div>
  </div>
  <div class="section">
    <h3 class="title">Section 3</h3>
    <div class="content">Contenu de la section 3</div>
  </div>
  <script>
    const titles = document.querySelectorAll('.title');

    titles.forEach(title => {
      title.onclick = function () {
        const content = this.nextElementSibling;

        // Fermer tous les autres contenus
```

```
        document.querySelectorAll('.content').forEach(c => {
          if (c !== content) {
            c.style.display = 'none';
          }
        });
        // Alterner l'affichage du contenu
        content.style.display = content.style.display === 'block' ? 'none' : 'block';
      };
    });
  </script>
</body>
</html>
```

Exercice 4: Développer un composant réutilisable

1. Créer un carousel ou accordéon réutilisable en encapsulant sa logique dans une fonction.
2. Fournir une documentation avec les instructions pour intégrer et personnaliser le composant.
3. Tester en intégrant plusieurs instances sur une même page.

Correction (Documentation et exemple dans GitHub)

- Voir l'exemple précédent en encapsulant les fonctions dans des méthodes exportables. Créer un `README.md` pour expliquer son intégration.

Développement et documentation d'un composant réutilisable

Voici une solution complète pour développer un **Carousel réutilisable**, accompagné de sa documentation.

Étape 1 : Création du composant Carousel

HTML : Structure de base.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
  <title>Carousel Component</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div id="carousel-container">
    <div id="carousel">
      <img id="carousel-image" src="" alt="Carousel Image">
      <button id="prev-button">Prev</button>
      <button id="next-button">Next</button>
    </div>
  </div>
  <script src="carousel.js"></script>
</body>
</html>
```

CSS : Mise en forme du Carousel.

```
/* style.css */
#carousel-container {
  width: 100%;
  max-width: 600px;
  margin: 0 auto;
  text-align: center;
}

#carousel {
  position: relative;
  display: flex;
  align-items: center;
  justify-content: center;
}
#carousel-image {
  width: 100%;
  height: auto;
  border: 2px solid #ccc;
  border-radius: 5px;
}
#prev-button, #next-button {
  position: absolute;
  top: 50%;
  transform: translateY(-50%);
  background-color: #007BFF;
  color: white;
  border: none;
  padding: 10px;
  cursor: pointer;
}

#prev-button {
```

```
  left: 10px;
}

#next-button {
  right: 10px;
}
```

JavaScript : Fonctionnement du Carousel.

```
// carousel.js
class Carousel {
  constructor(containerId, imageUrls) {
    this.container = document.getElementById(containerId);
    this.imageElement = this.container.querySelector('#carousel-image');
    this.prevButton = this.container.querySelector('#prev-button');
    this.nextButton = this.container.querySelector('#next-button');
    this.images = imageUrls;
    this.currentIndex = 0;
    // Initialize the carousel
    this.showImage(this.currentIndex);
    // Event Listeners
    this.prevButton.addEventListener('click', () => this.showPrevImage());
    this.nextButton.addEventListener('click', () => this.showNextImage());
  }

  showImage(index) {
    this.imageElement.src = this.images[index];
  }
  showPrevImage() {
    this.currentIndex = (this.currentIndex - 1 + this.images.length) % this.images.length;
    this.showImage(this.currentIndex);
  }
  showNextImage() {
    this.currentIndex = (this.currentIndex + 1) % this.images.length;
    this.showImage(this.currentIndex);
  }
}
// Usage example
document.addEventListener('DOMContentLoaded', () => {
  const images = [
    'https://via.placeholder.com/600x300/FF0000/FFFFFF?text=Image+1',
    'https://via.placeholder.com/600x300/00FF00/FFFFFF?text=Image+2',
    'https://via.placeholder.com/600x300/0000FF/FFFFFF?text=Image+3'
  ];
  new Carousel('carousel', images);
});
```

Étape 2 : Documentation du composant

Page de documentation :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Carousel Documentation</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Carousel Component Documentation</h1>
  <h2>Description</h2>
  <p>
    Le composant <strong>Carousel</strong> permet d'afficher une galerie d'images.
    L'utilisateur peut naviguer entre les images grâce aux boutons "Prev" et "Next".
  </p>

  <h2>Fonctionnalités</h2>
  <ul>
    <li>Affichage d'une image à la fois</li>
    <li>Boutons pour naviguer vers l'image précédente ou suivante</li>
    <li>Prise en charge de listes d'images dynamiques</li>
  </ul>

  <h2>Instructions d'intégration</h2>
  <ol>
    <li>Inclure les fichiers CSS et JS dans votre projet.</li>
    <li>Créer une structure HTML contenant un conteneur avec un ID unique.</li>
    <li>Appeler la classe <code>Carousel</code> en lui passant l'ID du conteneur et un
tableau d'URLs d'images.</li>
  </ol>

  <h3>Exemple d'intégration</h3>
  <pre>
<xmp>
<div id="carousel-container">
  <div id="carousel">
    <img id="carousel-image" src="" alt="Carousel Image">
    <button id="prev-button">Prev</button>
    <button id="next-button">Next</button>
  </div>
</div>

<script>
  const images = ['image1.jpg', 'image2.jpg', 'image3.jpg'];
  new Carousel('carousel', images);
</script>
```

```
</xmp>
  </pre>

  <h2>Compatibilité</h2>
  <p>
    Compatible avec les navigateurs modernes : Chrome, Firefox, Edge, Safari.
  </p>
</body>
</html>
```

Étape 3 : Demo

```
<!-- demo ex4 -->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script src="carousel.js"></script>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div id="carousel-container">
    <div id="carousel">
      <img id="carousel-image" src="" alt="Carousel Image">
      <button id="prev-button">Prev</button>
      <button id="next-button">Next</button>
    </div>
  </div>
  <script>
    const images = ['./images/tunisia1.jpg', './images/tunisia2.jpg', './images/tunisia3.jpg'];

    document.addEventListener('DOMContentLoaded', () => {
      new Carousel('carousel', images);
    });
  </script>
</body>
</html>
```

Vérifications

1. **Réutilisabilité** :
    - Modifier le tableau `images` pour tester avec d'autres URLs.
    - Ajouter plusieurs instances du composant sur une même page.
2. **Personnalisation** :
    - Adapter les styles avec le fichier CSS.

    - Ajouter des fonctionnalités supplémentaires (par ex. un minuteur pour passer automatiquement les images).
3. **Publication** :
    - Héberger les fichiers sur GitHub pour le partage.
    - Inclure la documentation et une démonstration en ligne.

Résultat attendu

- Une page fonctionnelle avec un **carousel interactif**.
- Une documentation claire pour les développeurs souhaitant intégrer le composant.